19

Support-Vector Machines and Kernel Methods

While linear logistic regression has been the mainstay in biostatistics and epidemiology, it has had a mixed reception in the machine-learning community. There the goal is often classification accuracy, rather than statistical inference. Logistic regression builds a classifier in two steps: fit a conditional probability model for $\Pr(Y = 1|X = x)$, and then classify as a one if $\widehat{\Pr}(Y = 1|X = x) \geq 0.5$. SVMs bypass the first step, and build a classifier directly.

Another rather awkward issue with logistic regression is that it fails if the training data are linearly separable! What this means is that, in the feature space, one can separate the two classes by a linear boundary. In cases such as this, maximum likelihood fails and some parameters march off to infinity. While this might have seemed an unlikely scenario to the early users of logistic regression, it becomes almost a certainty with modern *wide* genomics data. When $p \gg n$ (more features than observations), we can typically always find a separating hyperplane. Finding an *optimal separating hyperplane* was in fact the launching point for SVMs. As we will see, they have more than this to offer, and in fact live comfortably alongside logistic regression.

SVMs pursued an age-old approach in statistics, of enriching the feature space through nonlinear transformations and basis expansions; a classical example being augmenting a linear regression with interaction terms. A linear model in the enlarged space leads to a nonlinear model in the ambient space. This is typically achieved via the "kernel trick," which allows the computations to be performed in the n-dimensional space for an arbitrary number of predictors p. As the field matured, it became clear that in fact this kernel trick amounted to estimation in a reproducing-kernel Hilbert space.

Finally, we contrast the kernel approach in SVMs with the nonparametric regression techniques known as kernel smoothing.

19.1 Optimal Separating Hyperplane

Figure 19.1 shows a small sample of points in $\mathbb{R}^2$, each belonging to one of two classes (blue or orange). Numerically we would score these classes as +1 for say blue, and -1 for orange.[1] We define a two-class linear classifier via a function $f(x) = \beta_0 + x'\beta$, with the convention that we classify a point x_0 as +1 if $f(x_0) > 0$, and as -1 if $f(x_0) < 0$ (on the fence we flip a coin). Hence the classifier itself is $C(x) = \text{sign}[f(x)]$. The decision

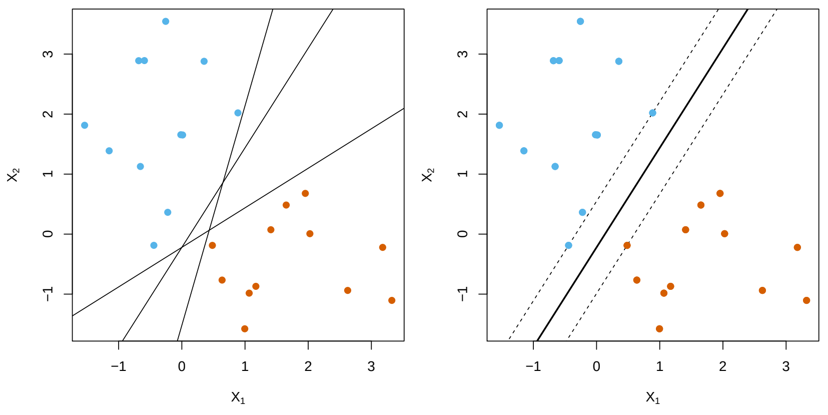

Figure 19.1 Left panel: data in two classes in $\mathbb{R}^2$. Three potential decision boundaries are shown; each separate the data perfectly. Right panel: the optimal separating hyperplane (a line in $\mathbb{R}^2$) creates the biggest margin between the two classes.

boundary is the set $\{x \mid f(x) = 0\}$. We see three different classifiers in the left panel of Figure 19.1, and they all classifier the points perfectly. The optimal separating hyperplane is the linear classifier that creates the largest *margin* between the two classes, and is shown in the right panel (it is also known as an optimal-margin classifier). The underlying hope is that, by making a big margin on the training data, it will also classify future observations well.

†1 Some elementary geometry† shows that the (signed) Euclidean distance from a point x_0 to the linear decision boundary defined by f is given by

$$\frac{1}{\|\beta\|_2} f(x_0). \tag{19.1}$$

With this in mind, for a separating hyperplane the quantity $\frac{1}{\|\beta\|_2} y_i f(x_i)$ is

[1] In this chapter, the ± 1 scoring leads to convenient notation.

the distance of x_i from the decision boundary.[2] This leads to an optimization problem for creating the optimal margin classifier:

$$\begin{aligned} &\underset{\beta_0,\,\beta}{\text{maximize}}\ M \\ &\text{subject to } \frac{1}{\|\beta\|_2} y_i(\beta_0 + x'\beta) \geq M,\ i = 1, \dots, n. \end{aligned} \tag{19.2}$$

A rescaling argument reduces this to the simpler form

$$\begin{aligned} &\underset{\beta_0,\,\beta}{\text{minimize}}\ \|\beta\|_2 \\ &\text{subject to } y_i(\beta_0 + x'\beta) \geq 1,\ i = 1, \dots, n. \end{aligned} \tag{19.3}$$

This is a quadratic program, which can be solved by standard techniques in convex optimization.† One noteworthy property of the solution is that †2

$$\hat{\beta} = \sum_{i \in \mathcal{S}} \hat{\alpha}_i x_i, \tag{19.4}$$

where $\mathcal{S}$ is the *support set*. We can see in Figure 19.1 that the margin touches three points (vectors); in this case there are $|\mathcal{S}| = 3$ support vectors, and clearly the orientation of $\hat{\beta}$ is determined by them. However, we still have to solve the optimization problem to identify the three points in $\mathcal{S}$, and their coefficients $\alpha_i,\ i \in \mathcal{S}$. Figure 19.2 shows an optimal-margin classifier fit to *wide* data, that is data where $p \gg n$. These are gene-expression measurements on $p = 3571$ genes measured on blood samples from $n = 72$ leukemia patients (first seen in Chapter 1). They were classified into two classes, 47 acute lymphoblastic leukemia (**ALL**) and 25 myeloid leukemia (**AML**). In cases like this, we are typically guaranteed a separating hyperplane[3]. In this case 42 of the 72 points are support points. One might be justified in thinking that this solution is overfit to this small amount of data. Indeed, when broken into a training and test set, we see that the test data encroaches well into the margin region, but in this case none are misclassified. Such classifiers are very popular in the wide-data world of genomics, largely because they seem to work very well. They offer a simple alternative to logistic regression, in a situation where the latter fails. However, sometimes the solution is overfit, and a modification is called for. This same modification takes care of nonseparable situations as well.

[2] Since all the points are correctly classified, the sign of $f(x_i)$ agrees with y_i, hence this quantity is always positive.

[3] If $n \leq p + 1$ we can always find a separating hyperplane, unless there are exact feature ties across the class barrier!

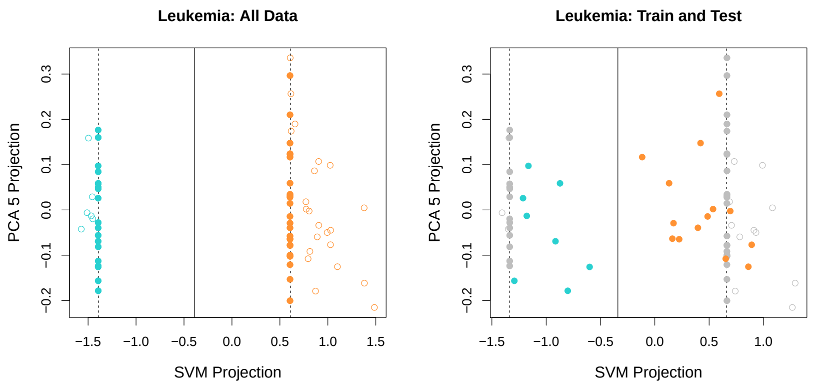

Figure 19.2 Left panel: optimal margin classifier fit to **leukemia** data. There are 72 observations from two classes—47 **ALL** and 25 **AML**—and 3571 gene-expression variables. Of the 72 observations, 42 are support vectors, sitting on the margin. The points are plotted against their fitted classifier function $\hat{f}(x)$, labeled SVM projection, and the fifth principal component of the data (chosen for display purposes, since it has low correlation with the former). Right panel: here the optimal margin classifier was fit to a random subset of 50 of the 72 observations, and then used to classify the remaining 22 (shown in color). Although these points fall on the wrong sides of their respective margins, they are all correctly classified.

19.2 Soft-Margin Classifier

Figure 19.3 shows data in $\mathbb{R}^2$ that are not separable. The generalization to a *soft* margin allows points to violate their margin. Each of the violators has a line segment connecting it to its margin, showing the extent of the violation. The soft-margin classifier solves

$$\begin{aligned} &\underset{\beta_0,\,\beta}{\text{minimize}}\ \|\beta\|_2 \\ &\text{subject to } y_i(\beta_0 + x_i'\beta) \geq 1 - \epsilon_i, \\ &\epsilon_i \geq 0,\ i = 1,\dots,n,\ \text{and } \sum_{i=1}^{n} \epsilon_i \leq B. \end{aligned} \tag{19.5}$$

Here B is the budget for the total amount of overlap. Once again, the solution has the form (19.4), except now the support set $\mathcal{S}$ includes any vectors on the margin as well as those that violate the margin. The bigger B, the

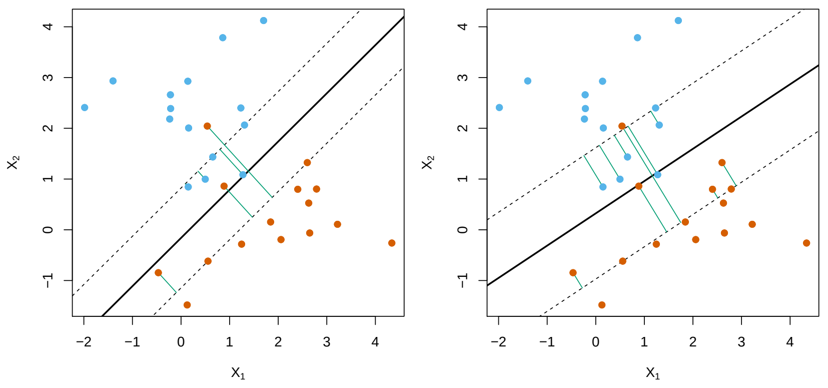

Figure 19.3 For data that are not separable, such as here, the soft-margin classifier allows margin violations. The budget B for the total measure of violation becomes a tuning parameter. The bigger the budget, the wider the soft margin and the more support points there are involved in the fit.

bigger the support set, and hence the more points that have a say in the solution. Hence bigger B means more stability and lower variance. In fact, even for separable data, allowing margin violations via B lets us regularize the solution by tuning B.

19.3 SVM Criterion as Loss Plus Penalty

It turns out that one can reformulate (19.5) and (19.3) in more traditional terms as the minimization of a loss plus a penalty:

$$\underset{\beta_0,\,\beta}{\text{minimize}} \sum_{i=1}^{n} [1 - y_i(\beta_0 + x_i'\beta)]_+ + \lambda\|\beta\|_2^2. \tag{19.6}$$

Here the *hinge* loss $L_H(y, f(x)) = [1 - yf(x)]_+$ operates on the margin quantity $yf(x)$, and is piecewise linear as in Figure 19.4.[†]The same margin †3 quantity came up in boosting in Section 17.4. The quantity $[1 - y_i(\beta_0 + x_i'\beta)]_+$ is the cost for x_i being on the wrong side of its margin (the cost is zero if it's on the correct side). The correspondence between (19.6) and (19.5) is exact; large λ corresponds to large B, and this formulation makes explicit the form of regularization. For separable data, the optimal separating hyperplane solution (19.3) corresponds to the limiting minimum-norm solution as $\lambda \downarrow 0$. One can show that the population minimizer of the

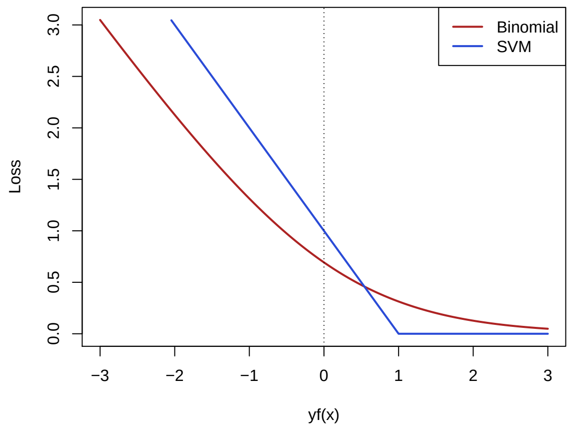

Figure 19.4 The hinge loss penalizes observation margins $yf(x)$ less than $+1$ linearly, and is indifferent to margins greater than $+1$. The negative binomial log-likelihood (deviance) has the same asymptotes, but operates in a smoother fashion near the *elbow* at $yf(x) = 1$.

hinge loss is in fact the Bayes classifier.[4] This shows that the SVM is in
†4 fact directly estimating the classifier $C(x) \in \{-1, +1\}$.†

The red curve in Figure 19.4 is (half) the binomial deviance for logistic regression (i.e. $f(x) = \beta_0 + x'\beta$ is now modeling logit $\Pr(Y = +1|X = x)$). With $Y = \pm 1$, the deviance can also be written in terms of the margin, and the ridged logistic regression corresponding to (19.6) has the form

$$\underset{\beta_0,\,\beta}{\text{minimize}} \sum_{i=1}^{n} \log[1 + e^{-y_i(\beta_0 + x_i'\beta)}] + \lambda\|\beta\|_2^2. \tag{19.7}$$

Logistic regression is discussed in Section 8.1, as well as Sections 16.5 and 17.4. This form of the binomial deviance is derived in (17.13) on page 343. These loss functions have some features in common, as can be seen in the figure. The binomial loss asymptotes to zero for large positive margins, and to a linear loss for large negative margins, matching the hinge loss in this regard. The main difference is that the hinge has a sharp elbow at +1, while the binomial bends smoothly. A consequence of this is that the binomial solution involves all the data, via weights $p_i(1 - p_i)$ that fade smoothly with distance from the decision boundary, as apposed to the binary nature

[4] The Bayes classifier $C(x)$ for a two-class problem using equal costs for misclassification errors assigns x to the class for which $\Pr(y|x)$ is largest.

of support points. Also, as seen in Section 17.4 as well, the population minimizer of the binomial deviance is the logit of the class probability

$$\lambda(x) = \log\left(\frac{\Pr(y = +1|x)}{\Pr(y = -1|x)}\right), \tag{19.8}$$

while that of the hinge loss is its sign $C(x) = \text{sign}[\lambda(x)]$. Interestingly, as $\lambda \downarrow 0$ the solution direction $\hat{\beta}$ to the ridged logistic regression problem (19.7) converges to that of the SVM.† †5

These forms immediately suggest other generalizations of the linear SVM. In particular, we can replace the ridge penalty $\|\beta\|_2^2$ by the sparsity-inducing lasso penalty $\|\beta\|_1$, which will set some coefficients to zero and hence perform feature selection. Publicly available software (e.g. package **`liblineaR`** in **`R`**) is available for fitting such lasso-regularized support-vector classifiers.

19.4 Computations and the Kernel Trick

The form of the solution $\hat{\beta} = \sum_{i \in \mathcal{S}} \hat{\alpha}_i x_i$ for the optimal- and soft-margin classifier has some important consequences. For starters, we can write the fitted function evaluated at a point x as

$$\begin{aligned} \hat{f}(x) &= \hat{\beta}_0 + x'\hat{\beta} \\ &= \hat{\beta}_0 + \sum_{i \in \mathcal{S}} \hat{\alpha}_i \langle x, x_i \rangle, \end{aligned} \tag{19.9}$$

where we have deliberately replaced the transpose notation with the more suggestive inner product. Furthermore, we show in (19.23) in Section 19.9 that the Lagrange dual involves the data only through the n^2 pairwise inner products $\langle x_i, x_j \rangle$ (the elements of the $n \times n$ *gram* matrix $\boldsymbol{X}\boldsymbol{X}'$). This means that the computations for computing the SVM solution scale linearly with p, although potentially cubic[5] in n. With very large p (in the tens of thousands and even millions as we will see), this can be convenient.

It turns out that all ridge-regularized linear models with wide data can be reparametrized in this way. Take ridge regression, for example:

$$\underset{\beta}{\text{minimize}} \; \|\boldsymbol{y} - \boldsymbol{X}\beta\|_2^2 + \lambda\|\beta\|_2^2. \tag{19.10}$$

This has solution $\hat{\beta} = (\boldsymbol{X}'\boldsymbol{X} + \lambda \boldsymbol{I}_p)^{-1}\boldsymbol{X}'\boldsymbol{y}$, and with p large requires inversion of a $p \times p$ matrix. However, it can be shown that $\hat{\beta} = \boldsymbol{X}'\hat{\alpha} =$

[5] In practice $O(n^2|\mathcal{S}|)$, and, with modern approximate solutions, much faster than that.

$\sum_{i=1}^n \hat{\alpha}_i x_i$, with $\hat{\alpha} = (\boldsymbol{X}\boldsymbol{X}' + \lambda \boldsymbol{I}_n)^{-1}\boldsymbol{y}$, which means the solution can be obtained in $O(n^2 p)$ rather than $O(np^2)$ computations. Again the gram matrix has played a role, and $\hat{\beta}$ has the same form as for the SVM.† †6

We now imagine expanding the p-dimensional feature vector x into a potentially much larger set $h(x) = [h_1(x), h_2(x), \ldots, h_m(x)]$; for an example to latch onto, think polynomial basis of total degree d. As long as we have an efficient way to compute the inner products $\langle h(x), h(x_j)\rangle$ for any x, we can compute the SVM solution in this enlarged space just as easily as in the original. It turns out that convenient *kernel* functions exist that do just that. For example $K_d(x,z) = (1 + \langle x, z\rangle)^d$ creates a basis expansion h_d of polynomials of total degree d, and $K_d(x,z) = \langle h_d(x), h_d(z)\rangle$.† †7

The polynomial kernels are mainly useful as existence proofs; in practice other more useful kernels are used. Probably the most popular is the radial kernel

$$K(x,z) = e^{-\gamma\|x-z\|_2^2}. \tag{19.11}$$

This is a positive definite function, and can be thought of as computing an inner product in some feature space. Here the feature space is in principle infinite-dimensional, but of course effectively finite.[6] Now one can think of the representation (19.9) in a different light;

$$\hat{f}(x) = \hat{\alpha}_0 + \sum_{i \in \mathcal{S}} \hat{\alpha}_i K(x, x_i), \tag{19.12}$$

an expansion of radial basis functions, each centered on one of the training examples. Figure 19.5 illustrates such an expansion in $\mathbb{R}^1$. Using such nonlinear kernels expands the scope of SVMs considerably, allowing one to fit classifiers with nonlinear decision boundaries.

One may ask what objective is being optimized when we move to this kernel representation. This is covered in the next section, but as a sneak preview we present the criterion

$$\underset{\alpha_0,\,\alpha}{\text{minimize}} \sum_{j=1}^n \left[1 - y_j\left(\alpha_0 + \sum_{i=1}^n \alpha_i K(x_j, x_i)\right)\right]_+ + \lambda \alpha' \boldsymbol{K} \alpha, \tag{19.13}$$

where the $n \times n$ matrix $\boldsymbol{K}$ has entries $K(x_j, x_i)$.

As an illustrative example in $\mathbb{R}^2$ (so we can visualize the nonlinear boundaries), we generated the data in Figure 19.6. We show two SVM

[6] A bivariate function $K(x,z)$ ($\mathbb{R}^p \times \mathbb{R}^p \mapsto \mathbb{R}^1$) is positive-definite if, for every q, every $q \times q$ matrix $\boldsymbol{K} = \{K(x_i, x_j)\}$ formed using distinct entries $x_1, x_2, \ldots, x_q$ is positive definite. The feature space is defined in terms of the eigen-functions of the kernel.

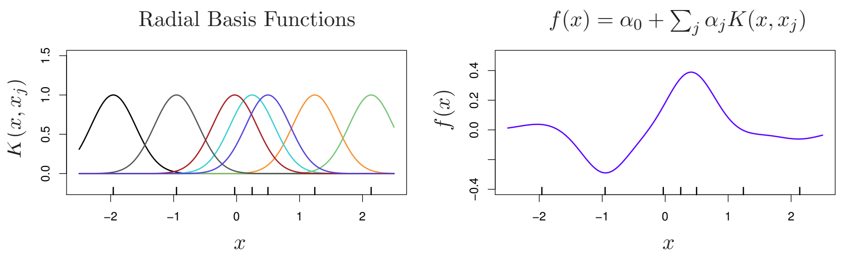

Figure 19.5 Radial basis functions in $\mathbb{R}^1$. The left panel shows a collection of radial basis functions, each centered on one of the seven observations. The right panel shows a function obtained from a particular linear expansion of these basis functions.

solutions, both using a radial kernel. In the left panel, some margin errors are committed, but the solution looks reasonable. However, with the flexibility of the enlarged feature space, by decreasing the budget B we can typically overfit the training data, as is the case in the right panel. A separate little blue island was created to accommodate the one blue point in a sea of brown.

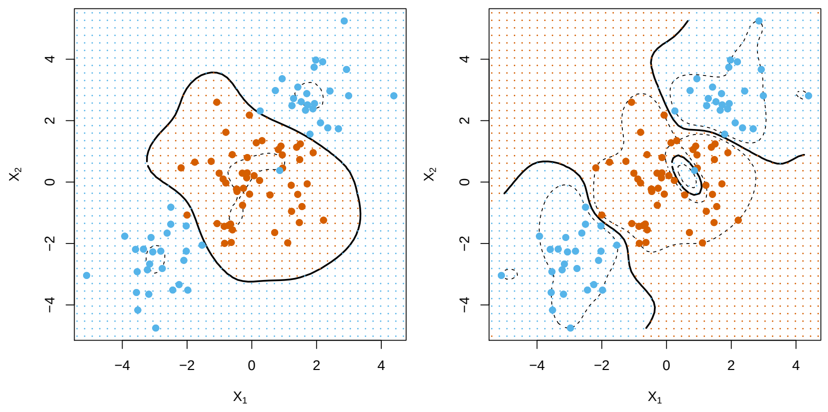

Figure 19.6 Simulated data in two classes in $\mathbb{R}^2$, with SVM classifiers computed using the radial kernel (19.11). The left panel uses a larger value of B than the right. The solid lines are the decision boundaries in the original space (linear boundaries in the expanded feature space). The dashed lines are the projected margins in both cases.

19.5 Function Fitting Using Kernels

The analysis in the previous section is heuristic—replacing inner products by kernels that compute inner products in some (implicit) feature space. Indeed, this is how kernels were first introduced in the SVM world. There is however a rich literature behind such approaches, which goes by the name *function fitting in reproducing-kernel Hilbert spaces (RKHSs)*. We give a very brief overview here. One starts with a bivariate positive-definite kernel $K : \mathbb{R}^p \times \mathbb{R}^p \rightarrow \mathbb{R}^1$, and we consider a space $\mathcal{H}_K$ of functions $f : \mathbb{R}^p \rightarrow \mathbb{R}^1$ generated by the kernel: $f \in \text{span}\{K(\cdot, z),\ z \in \mathbb{R}^p\}$[7] The
†8 kernel also induces a norm on the space $\|f\|_{\mathcal{H}_K}$,† which can be thought of as a roughness measure.

We can now state a very general optimization problem for fitting a function to data, when restricted to this class;

$$\underset{f \in \mathcal{H}_K}{\text{minimize}} \left\{ \sum_{i=1}^{n} L(y_i, \alpha_0 + f(x_i)) + \lambda \|f\|^2_{\mathcal{H}_K} \right\}, \tag{19.14}$$

a search over a possibly infinite-dimensional function space. Here L is an arbitrary loss function. The "magic" of these spaces in the context of this problem is that one can show that the solution is finite-dimensional:

$$\hat{f}(x) = \sum_{i=1}^{n} \hat{\alpha}_i K(x, x_i), \tag{19.15}$$

a linear basis expansion with basis functions $k_i(x) = K(x, x_i)$ anchored at each of the observed "vectors" x_i in the training data. Moreover, using the "reproducing" property of the kernel in this space, one can show that the penalty reduces to

$$\|\hat{f}\|^2_{\mathcal{H}_K} = \sum_{i=1}^{n} \sum_{j=1}^{n} \hat{\alpha}_i \hat{\alpha}_j K(x_i, x_j) = \hat{\alpha}' \boldsymbol{K} \hat{\alpha}. \tag{19.16}$$

Here $\boldsymbol{K}$ is the $n \times n$ *gram* matrix of evaluations of the kernel, equivalent to the $\boldsymbol{XX}'$ matrix for the linear case.

Hence the abstract problem (19.14) reduces to the generalized ridge problem

$$\underset{\alpha \in \mathbb{R}^n}{\text{minimize}} \left\{ \sum_{i=1}^{n} L\left(y_i, \alpha_0 + \sum_{j=1}^{n} \alpha_i K(x_i, x_j) \right) + \lambda \alpha' \boldsymbol{K} \alpha \right\}. \tag{19.17}$$

[7] Here $k_z = K(\cdot, z)$ is considered a function of the first argument, and the second argument is a parameter.

Indeed, if L is the hinge loss as in (19.6), this is the equivalent "loss plus penalty" criterion being fit by the kernel SVM. Alternatively, if L is the binomial deviance loss as in (19.7), this would fit a kernel version of logistic regression. Hence most fitting methods can be generalized to accommodate kernels.

This formalization opens the door to a wide variety of applications, depending on the kernel function used. Alternatively, as long as we can compute suitable similarities between objects, we can build sophisticated classifiers and other models for making predictions about other attributes of the objects.[8] In the next section we consider a particular example.

19.6 Example: String Kernels for Protein Classification

One of the important problems in computational biology is to classify proteins into functional and structural classes based on their sequence similarities. Protein molecules can be thought of as strings of amino acids, and differ in terms of length and composition. In the example we consider, the lengths vary between 75 and 160 amino-acid molecules, each of which can be one of 20 different types, labeled using the letters of the alphabet.

Here follow two protein examples x_1 and x_2, of length 110 and 153 respectively:

IPTSALVKETLALLSTHRTLLIANETLRIPVPVHKNHQLCTEEIFQGIGTLESQTVQGGTV
ERLFKNLSLIKKYIDGQKKKCGEERRRVNQFLDY**LQE**FLGVMNTEWI

PHRRDLCSRSIWLARKIRSDLTALTESYVKHQGLWSELTEAER**LQE**NLQAYRTFHVLLA
RLLEDQQVHFTPTEGDFHQAIHTLLLQVAAFAYQIEELMILLEYKIPRNEADGMLFEKK
LWGLKV**LQE**LSQWTVRSIHDLRFISSHQTGIP

We treat the proteins x as documents consisting of letters, with a dictionary of size 20. Our feature vector $h^m(x)$ will consist of the counts for all m-grams in the protein—that is, distinct sequences of consecutive letters of length m. As an illustration, we use $m = 3$, which results in 20^3 =8,000 possible sub-sequences; hence $h^3(x)$ will be a vector of length 8,000, with each element the number of times that particular sub-sequence occurs in the protein x. In our example, the sub-sequence **LQE** occurs once in the first, and twice in the second protein, so $h^3_{\text{LQE}}(x_1) = 1$ and $h^3_{\text{LQE}}(x_2) = 2$.

The number of possible sequences of length m is 20^m, which can be very

[8] As long as the similarities behave like inner products; i.e. they form positive semi-definite matrices.

large for moderate m. Also the vast majority of the sub-sequences do not match the strings in our training set, which means $h^m(x)$ will be sparse. It turns out that we can compute the $n \times n$ inner product matrix or *string kernel* $\boldsymbol{K}_m(x_1, x_2) = \langle h^m(x_1), h^m(x_2)\rangle$ efficiently using tree structures, without †9 actually computing the individual vectors.† Armed with the kernel, we

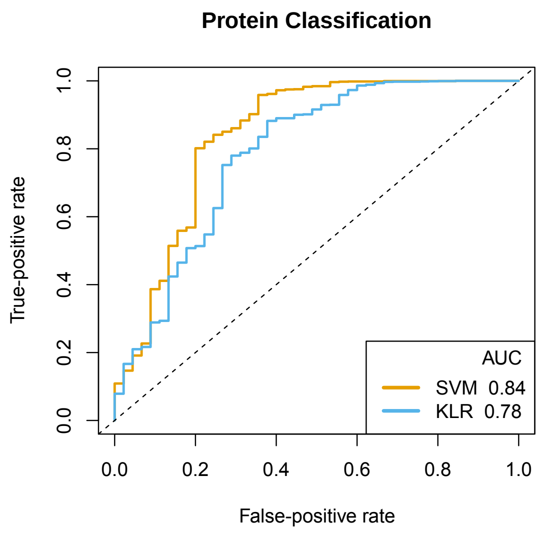

Figure 19.7 ROC curves for two classifiers fit to the protein data. The ROC curves were computed using 10-fold cross-validation, and trace the trade-off between false-positive and true-positive error rates as the classifier threshold is varied. The area under the curve (AUC) summarizes the overall performance of each classifier. Here the SVM is slightly superior to kernel logistic regression.

can now use it to fit a regularized SVM or logistic regression model, as outlined in the previous section. The data consist of 1708 proteins in two classes—negative (1663) and positive (45). We fit both the kernel SVM and kernel logistic regression models. For both methods, cross-validation suggested a very small value for λ. Figure 19.7 shows the ROC trade-off curve for each, using 10-fold cross-validation. Here the SVM outperforms logistic regression.

19.7 SVMs: Concluding Remarks

SVMs have been wildly successful, and are one of the “must have” tools in any machine-learning toolbox. They have been extended to cover many different scenarios, other than two-class classification, with some awkwardness in cases. The extension to nonlinear function-fitting via kernels (inspiring the “machine” in the name) generated a mini industry. Kernels are parametrized, learned from data, with special problem-specific structure, and so on.

On the other hand, we know that fitting high-dimensional nonlinear functions is intrinsically difficult (the “curse of dimensionality”), and SVMs are not immune. The quadratic penalty implicit in kernel methodology means all features are included in the model, and hence sparsity is generally not an option. Why then this unbridled enthusiasm? Classifiers are far less sensitive to bias–variance tradeoffs, and SVMs are mostly popular for their classification performance. The ability to define a kernel for measuring similarities between abstract objects, and then train a classifier, is a novelty added by these approaches that was missed in the past.

19.8 Kernel Smoothing and Local Regression

The phrase “kernel methodology” might mean something a little different to statisticians trained in the 1970–90 period. Kernel smoothing represents a broad range of tools for performing non- and semi-parametric regression. Figure 19.8 shows a Gaussian kernel smooth fit to some artificial data $\{x_i, y_i\}_1^n$. It computes at each point x_0 a weighted average of the y-values of neighboring points, with weights given by the height of the kernel. In its simplest form, this estimate can be written as

$$\hat{f}(x_0) = \sum_{i=1}^{n} y_i K_\gamma(x_0, x_i), \tag{19.18}$$

where $K_\gamma(x_0, x_i)$ represents the radial kernel with width parameter γ.[9] Notice the similarity to (19.15); here the $\hat{\alpha}_i = y_i$, and the complexity of the model is controlled by γ. Despite this similarity, and the use of the same kernel, these methodologies are rather different.

The focus here is on local estimation, and the kernel does the localizing. Expression (19.18) is almost a weighted average—almost because

[9] Here $K_\gamma(x, \mu)$ is the normalized Gaussian density with mean μ and variance $1/\gamma$.

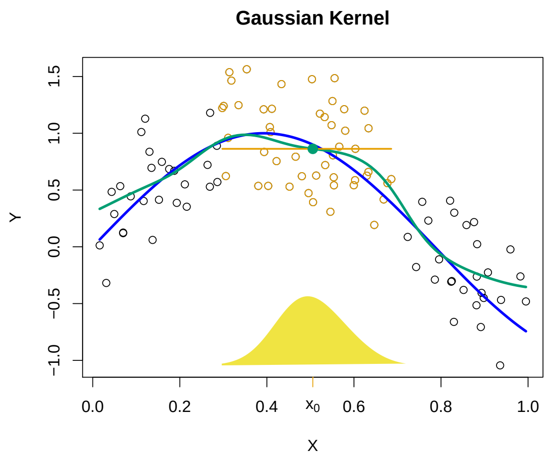

Figure 19.8 A Gaussian kernel smooth of simulated data. The points come from the blue curve with added random errors. The kernel smoother fits a weighted mean of the observations, with the weighting kernel centered at the target point, x_0 in this case. The points shaded orange contribute to the fit at x_0. As x_0 moves across the domain, the smoother traces out the green curve. The width of the kernel is a tuning parameter. We have depicted the Gaussian weighting kernel in this figure for illustration; in fact its vertical coordinates are all positive and integrate to one.

$\sum_{i=1}^{n} K_\gamma(x_0, x_i) \approx 1$. In fact, the Nadaraya–Watson estimator is more explicit:

$$\hat{f}_{NW}(x_0) = \frac{\sum_{i=1}^{n} y_i K_\gamma(x_0, x_i)}{\sum_{i=1}^{n} K_\gamma(x_0, x_i)}. \tag{19.19}$$

Although Figure 19.8 is one-dimensional, the same formulation applies to x in higher dimensions.

Weighting kernels other than the Gaussian are typically favored; in particular, near-neighbor kernels with compact support. For example, the tricube kernel used by the `lowess` smoother in **R** is defined as follows:

1 Define $d_i = \|x_0 - x_i\|_2,\ i = 1, \ldots, n$, and let $d_{(m)}$ be the mth smallest (the distance of the mth nearest neighbor to x_0). Let $u_i = d_i/d_{(m)},\ i = 1, \ldots, n$.

2 The tricube kernel is given by

$$K_s(x_0, x_i) = \begin{cases} \left(1 - u_i^3\right)^3 & \text{if } u_i \le 1; \\ 0 & \text{otherwise,} \end{cases} \tag{19.20}$$

where $s = m/n$, the *span* of the kernel. Near-neighbor kernels such as this adapt naturally to the local density of the x_i; wider in low-density regions, narrower in high-density regions. A tricube kernel is illustrated in Figure 19.9.

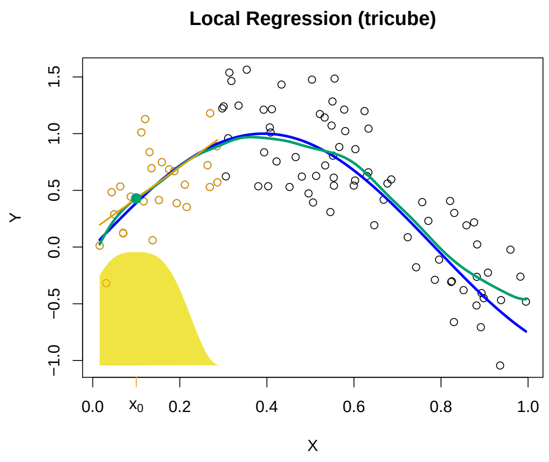

Figure 19.9 Local regression fit to the simulated data. At each point x_0, we fit a locally weighted linear least-squares model, and use the fitted value to estimate $\hat{f}_{LR}(x_0)$. Here we use the tricube kernel (19.20), with a span of 25%. The orange points are in the weighting neighborhood, and we see the orange linear fit computed by kernel weighted least squares. The green dot is the fitted value at x_0 from this local linear fit.

Weighted means suffer from boundary bias—we can see in Figure 19.8 that the estimate appears biased upwards at both boundaries. The reason is that, for example on the left, the estimate for the function on the boundary averages points always to the right, and since the function is locally increasing, there is an upward bias. *Local linear regression* is a natural generalization that fixes such problems. At each point x_0 we solve the following weighted least-squares problem

$$(\hat{\beta}_0(x_0), \hat{\beta}(x_0)) = \arg\min_{\beta_0,\beta} \sum_{i=1}^{n} K_s(x_0, x_i)(y_i - \beta_0 - x_i\beta)^2. \qquad (19.21)$$

Then $\hat{f}_{LR}(x_0) = \hat{\beta}_0(x_0) + x_0\hat{\beta}(x_0)$. One can show that, to first order,
†10 $\hat{f}_{LR}(x_0)$ removes the boundary bias exactly.†

Figure 19.9 illustrates the procedure on our simulated data, using the tricube kernel with a span of 25% of the data. In practice, the width of the kernel (the span here) has to be selected by some means; typically we use cross-validation.

Local regression works in any dimension; that is, we can fit two- or higher-dimensional surfaces using exactly the same technique. Here the ability to remove boundary bias really pays off, since the boundaries can be complex. These are referred to as *memory-based methods*, since there is no fitted model. We have to save all the training data, and recompute the local fit every time we make a prediction.

Like kernel SVMs and their relatives, kernel smoothing and local regression break down in high dimensions. Here the near neighborhoods become so wide that they are no longer local.

19.9 Notes and Details

In the late 1980s and early 1990s, machine-learning research was largely driven by prediction problems, and the neural-network community at AT&T Bell laboratories was amongst the leaders. The problem of the day was the US Post-Office handwritten zip-code OCR challenge—a 10-class image classification problem. Vladimir Vapnik was part of this team, and along with colleagues invented a more direct approach to classification, the support-vector machine. This started with the seminal paper by Boser *et al.* (1992), which introduced the optimal margin classifier (optimal separating hyperplane); see also Vapnik (1996). The ideas took off quite rapidly, attracting a large cohort of researchers, and evolved into the more general class of "kernel" methods—that is, models framed in reproducing-kernel Hilbert spaces. A good general reference is Schölkopf and Smola (2001).

†1 [p. 376] *Geometry of separating hyperplanes.* Let $f(x) = \beta' x + \beta_0$ define a linear decision boundary $\{x \mid f(x) = 0\}$ in $\mathbb{R}^p$ (an affine set of codimension one). The unit vector normal to the boundary is $\beta/\|\beta\|_2$, where $\|\cdot\|_2$ denotes the ℓ_2 or Euclidean norm. How should one compute the distance from a point x to this boundary? If x_0 is any point on the boundary

(i.e. $f(x_0) = 0$), we can project $x - x_0$ onto the normal, giving us

$$\frac{\beta'(x - x_0)}{\|\beta\|_2} = \frac{1}{\|\beta\|_2} f(x),$$

as claimed in (19.1). Note that this is the signed distance, since $f(x)$ will be positive or negative depending on what side of the boundary it lies on.

†2 [p. 377] *The "support" in SVM.* The Lagrange primal problem corresponding to (19.3) can be written as

$$\underset{\beta_0,\,\beta}{\text{minimize}} \left\{ \frac{1}{2}\beta'\beta + \sum_{i=1}^{n} \gamma_i [1 - y_i(\beta_0 + x_i'\beta)] \right\}, \tag{19.22}$$

where $\gamma_i \geq 0$ are the Lagrange multipliers. On differentiating we find that $\beta = \sum_{i=1}^{n} \gamma_i y_i x_i$ and $\sum_{i=1}^{n} y_i \gamma_i = 0$. With $\alpha_i = y_i \gamma_i$, we get (19.4), and note that the positivity constraint on γ_i will lead to some of the α_i being zero. Plugging into (19.22) we obtain the Lagrange dual problem

$$\begin{aligned} &\underset{\{\gamma_i\}_1^n}{\text{maximize}} \left\{ \sum_{i=1}^{n} \gamma_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \gamma_i \gamma_j y_i y_j x_i' x_j \right\} \\ &\qquad\qquad \text{subject to } \gamma_i \geq 0, \quad \sum_{i=1}^{n} y_i \gamma_i = 0. \end{aligned} \tag{19.23}$$

†3 [p. 379] *The SVM loss function.* The constraint in (19.5) can be succinctly captured via the expression

$$\sum_{i=1}^{n} [1 - y_i(\beta_0 + x_i'\beta)]_+ \leq B. \tag{19.24}$$

We only require a (positive) ϵ_i if our margin is less than 1, and we get charged for the sum of these ϵ_i. We now use a Lagrange multiplier to enforce the constraint, leading to

$$\underset{\beta_0,\,\beta}{\text{minimize}} \ \|\beta\|_2^2 + \gamma \sum_{i=1}^{n} [1 - y_i(\beta_0 + x_i'\beta)]_+. \tag{19.25}$$

Multiplying by $\lambda = 1/\gamma$ gives us (19.6).

†4 [p. 380] *The SVM estimates a classifier.* The following derivation is due to Wahba *et al.* (2000). Consider

$$\underset{f(x)}{\text{minimize}} \ E_{Y|X=x} \{[1 - Yf(x)]_+\}. \tag{19.26}$$

Dropping the dependence on x, the objective can be written as $P_+[1-f]_+ + P_-[1+f]_+$, where $P_+ = \Pr(Y=+1|X=x)$, and $P_- = \Pr(Y=-1|X=x) = 1-P_+$. From this we see that

$$f = \begin{cases} +1 & \text{if } P_+ > \frac{1}{2} \\ -1 & \text{if } P_- < \frac{1}{2}. \end{cases} \tag{19.27}$$

†5 [p. 381] *SVM and ridged logistic regression.* Rosset *et al.* (2004) show that the limiting solution as $\lambda \downarrow 0$ to (19.7) for separable data coincides with that of the SVM, in the sense that $\hat{\beta}/\|\hat{\beta}\|_2$ converges to the same quantity for the SVM. However, because of the required normalization for logistic regression, the SVM solution is preferable. On the other hand, for overlapped situations, the logistic-regression solution has some advantages, since its target is the logit of the class probabilities.

†6 [p. 382] *The kernel trick.* The trick here is to observe that from the score equations we have $-\boldsymbol{X}'(\boldsymbol{y}-\boldsymbol{X}\beta) + \lambda\beta = 0$, which means we can write $\hat{\beta} = \boldsymbol{X}'\alpha$ for some α. We now plug this into the score equations, and some simple manipulation gives the result. A similar result holds for ridged logistic regression, and in fact any linear model with a ridge penalty on the coefficients (Hastie and Tibshirani, 2004).

†7 [p. 382] *Polynomial kernels.* Consider $K_2(x,z) = (1+\langle x,z\rangle)^2$, for x (and z) in $\mathbb{R}^2$. Expanding we get

$$K_2(x,z) = 1 + 2x_1z_1 + 2x_2z_2 + 2x_1x_2z_1z_2 + x_1^2z_1^2 + x_2^2z_2^2.$$

This corresponds to $\langle h_2(x), h_2(z)\rangle$ with

$$h_2(x) = (1, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_1x_2, x_1^2, x_2^2).$$

The same is true for $p > 2$ and for degree $d > 2$.

†8 [p. 384] *Reproducing kernel Hilbert spaces.* Suppose K has eigen expansion $K(x,z) = \sum_{i=1}^{\infty} \gamma_i\phi_i(x)\phi_i(z)$, with $\gamma_i \geq 0$ and $\sum_{i=1}^{\infty}\gamma_i < \infty$. Then we say $f \in \mathcal{H}_K$ if $f(x) = \sum_{i=1}^{\infty} c_i\phi_i(x)$, with

$$\|f\|_{\mathcal{H}_K}^2 \equiv \sum_{i=1}^{\infty} \frac{c_i^2}{\gamma_i} < \infty. \tag{19.28}$$

Often $\|f\|_{\mathcal{H}_K}$ behaves like a roughness penalty, in that it penalizes unlikely members in the span of $K(\cdot, z)$ (assuming that these correspond to "rough" functions). If f has some high loadings c_j on functions ϕ_j with small eigenvalues γ_j (i.e. not prominent members of the span), the norm becomes large. Smoothing splines and their generalizations correspond to function fitting in a RKHS (Wahba, 1990).

†9 [p. 386] This methodology and the data we use in our example come from Leslie *et al.* (2003).

†10 [p. 390] *Local regression and bias reduction.* By expanding the unknown true $f(x)$ in a first-order Taylor expansion about the target point x_0, one can show that $E\hat{f}_{LR}(x_0) \approx f(x_0)$ (Hastie and Loader, 1993).